AFRL-RI-RS-TR-2018-073

PROMOTING PROBABILISTIC PROGRAMMING SYSTEM (PPS) DEVELOPMENT IN PROBABILISTIC PROGRAMMING FOR ADVANCING MACHINE LEARNING (PPAML)

GALOIS, INC.

MARCH 2018

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 88th ABW, Wright-Patterson AFB Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2018-073 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
STEVEN L. DRAGER
Work Unit Manager

/ S /
JOHN D. MATYJAS
Technical Advisor, Computing
& Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
MARCH 2018	FINAL TECHNICAL REPORT	OCT 2013 – OCT 2017

4. TITLE AND SUBTITLE

PROMOTING PROBABILISTIC PROGRAMMING SYSTEM (PPS) DEVELOPMENT IN PROBABILISTIC PROGRAMMING FOR ADVANCING MACHINE LEARNING (PPAML)

5a. CONTRACT NUMBER
FA8750-14-C-0003

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
61101E

6. AUTHOR(S)

Eric Woldridge

5d. PROJECT NUMBER
PPML

5e. TASK NUMBER
1G

5f. WORK UNIT NUMBER
AL

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Galois, Inc
421 SW 6th Ave Ste 300
Portland, OR 97204

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RITA
525 Brooks Road
Rome NY 13441-4505

DARPA
675 North Randolph Street
Arlington, VA 22203-2114

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2018-073

12. DISTRIBUTION AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited. PA# 88ABW-2018-1159
Date Cleared: 8 March 2018

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Machine Learning has demonstrated the potential to transform many areas of science, commerce, and the military. However, creating and maintaining successful machine learning systems is an arduous task that requires a doctoral degree and heroic software engineering efforts. Probabilistic Programming for Advancing Machine Learning (PPAML) — by creating probabilistic programming systems and associated solvers—aimed to make existing machine learning applications easier to build and to greatly extend the range of problems that can be successfully solved by machine learning.

This effort acted as the voice of the user: (a) exposing the probabilistic programming, machine learning and inference engine performers to a breadth of user scenarios over a wide a variety of domains, (b) evaluated and produced feedback on PPS tools to enable the performer teams to understand user perspectives and spur them to enhance their PPS for future users, and (c) developed a community of users in multiple distinct application areas who are invested in the future developments of PPSs.

15. SUBJECT TERMS

Probabilistic Programming, Machine Learning, Artificial Intelligence, Evaluation

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**STEVEN L. DRAGER**
U	U	U	UU	34	19b. TELEPHONE NUMBER (*Include area code*)

TABLE OF CONTENTS

Section **Page**

LIST OF FIGURES

1.0 SUMMARY

Machine learning is at the heart of modern approaches to artificial intelligence. The field posits that teaching computers how to learn can be significantly more effective than programming them explicitly. This idea has revolutionized what computers can do in a wide range of domains, including Intelligence, Surveillance, and Reconnaissance (ISR), Natural Language Processing (NLP), Predictive Analytics, Cyber, and various scientific disciplines. Example applications include self-driving cars, image search and activity detection, object tracking, topic models, spam filters, recommender systems, predictive databases, and gene sequencing. Unfortunately, building effective machine learning applications currently requires Herculean efforts on the part of highly trained experts in machine learning. Probabilistic Programming is a new programming paradigm for managing uncertain information.

The goal of the Probabilistic Programming for Advancing Machine Learning (PPAML) program is to facilitate the construction of machine learning applications by using probabilistic programming to: (1) dramatically increase the number of people who can successfully build machine learning applications; (2) make machine learning experts radically more effective; and (3) enable new applications that are inconceivable today.

As the Technical Area (TA) 1 (TA-1) Domain Experts team for PPAML, Galois contributed more than 20 years of experience developing high-level and domain-specific languages and more than 35 years of experience in application driven machine learning research.

2.0 INTRODUCTION

Machine Learning has demonstrated the potential to transform many areas of science, commerce, and the military. However, creating and maintaining successful machine learning systems is an arduous task that requires a doctoral degree and heroic software engineering efforts. One cause of this is the lack of good languages for defining machine learning models and systems. PPAML— by creating probabilistic programming systems and associated solvers—aimed to make existing machine learning applications easier to build and to greatly extend the range of problems that can be successfully solved by machine learning. Galois, as the TA-1 Domain Experts, also was tasked as the research evaluator and had the following areas of contribution during the program:

1. Develop Challenge Problems (CPs)
2. Evaluate Probabilistic Programming Systems (PPSs)
3. Organize and Run Summer Schools
4. Foster Collaboration
5. Evaluate Team Challenge Problems

As the TA-1 team we were the voice of the user. Over the course of the program we were able to (a) expose TA-2 Probabilistic Programming, TA-3 Machine Learning, and TA-4 Inference Engine, heretofore referred to as TA2-4 performers, to a breadth of user scenarios in a wide a variety of domains, (b) produce feedback on PPS tools that enable the TA2-4 teams to understand user perspectives and spur them to enhance their PPS for future users, and

(c) develop a community of users in multiple distinct application areas who are invested in the future developments of PPSs.

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

Section 3 describes the methods for each of the primary areas of contribution: Development of challenge problems, evaluation of TA2-4 PPSs and team challenge problems, annual summer schools and program collaboration. Results from these efforts are presented in section 4.

3.1 Develop Challenge Problems

The TA-1 team was responsible for developing and delivering challenge problems to performers in TA2-4. A total of ten challenge problems were developed and delivered over the course of the program. Three challenge problems were delivered at the program kick-off. Subsequently, the TA-1 team delivered a new challenge problem every six months, introduced at program Principal Investigator (PI) meetings.

A framework and set of problem characteristics were developed for evaluating and selecting candidate problems. We divided these into required characteristics and "diversity" characteristics. Required characteristics are properties that all candidate challenge problems must possess because they enable experimentation and the development of probabilistic solutions. The diversity characteristics were designed to ensure that over the course of the program the population of challenge problems covered a wide range of domains and problem types.

Required Characteristics:
Every challenge problem was required to have the following properties:

Learning: The problem involved learning a probabilistic model and then applying that model to perform some task. In some cases, we defined some pure inference sub-problems to evaluate the solvers under controlled conditions.

Mature Data: The data must be in a form that is ready to be modeled in a PPS, available in sufficient quantity, free for distribution, and engineered for use by modelers. We sought problems that had both high-level representations of data (via careful feature engineering) and low-level representations (near to sensor data).

Scalability: The problem supports the construction of interesting problem variants. Natural dimensions for scaling include the number of random variables (e.g., via changing the spatial or temporal extent or resolution of the data), and the amount of data (where both very small and very large data sets are of interest). Other scaling dimensions include introducing additional information sources (e.g., for information fusion problems) or multiple types of queries. These latter dimensions allowed us to test the degree to which the probabilistic programming languages support modularity and evolution by measuring how easy it is to modify the programs to incorporate additional information sources and queries.

Diversity Characteristics: A Taxonomy of Challenge Problems

To ensure generality of PPSs—and to address problems of interest to DARPA—the set of ten challenge problems selected over the course of the program covered a wide range of domains, data structures, model types, and queries. We developed the following taxonomy to assess these factors. Our taxonomy exposes the specific aspects of PPS tested by each problem.

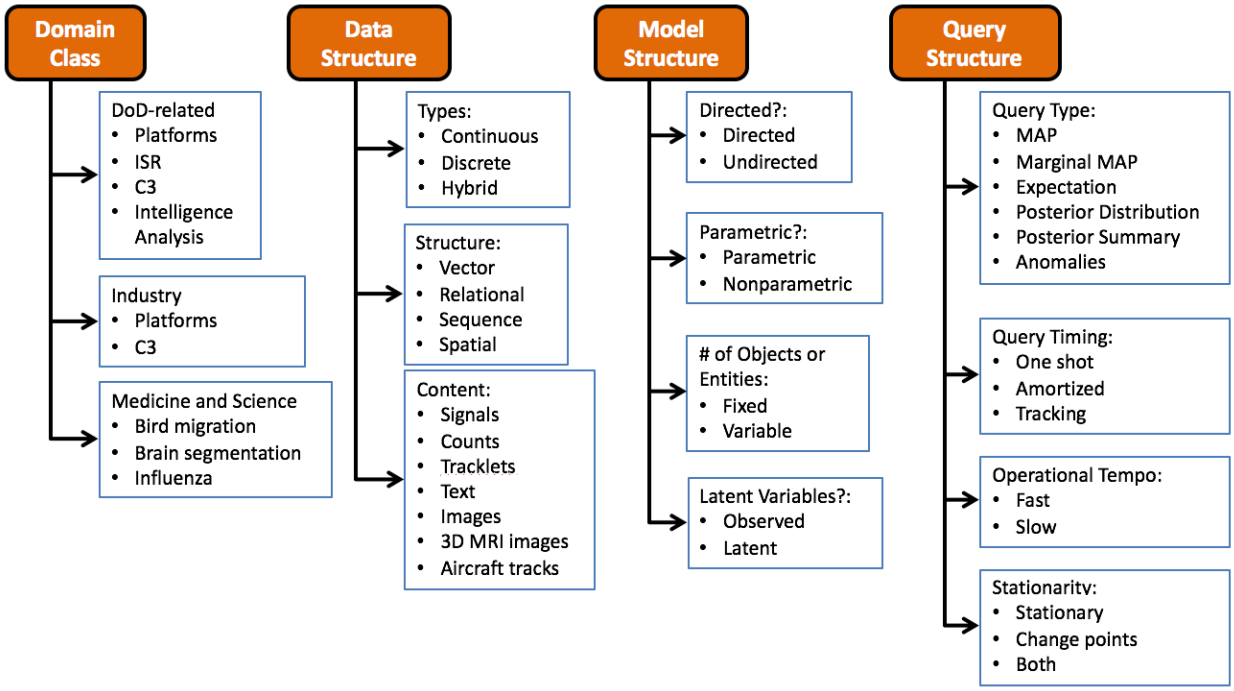

Figure 1 Challenge Problem Dimensions

Problem Domain: PPAML sought to provide new approaches for solving machine learning problems of interest to defense, science, and the economy. Within DoD, there are different needs for intelligence analysis; intelligence, surveillance, and reconnaissance; command, control, and communications; and management of individual platforms (system identification, localization, control, and health monitoring). Distinctive problems also arise in science (e.g., population modeling), medicine (e.g., brain structure modeling), and industry (e.g., engine health modeling).

The Data Structure: Different kinds of data naturally require different support within a PPS. We classified data structures according to content type (sensor signals, tracklets, counts, events, text, images, tabular data), data structure (vector data versus relational data), and data type (discrete, continuous, or a hybrid of the two). Many probabilistic modeling and inference methods are specialized to only discrete or only continuous data, and inference can become very challenging with hybrid data that mix discrete and continuous variables. Most machine learning methods focus on vector data. Relational data have typically required different approaches than vector data. Time series, spatial, and graph data are all special forms of relational data.

The Model Structure: A core issue for PPAML was to develop languages that can express a wide range of models. We classified model structures according to three dimensions: directed versus undirected, parametric versus non-parametric, and fixed versus a variable number of

objects. Classical Bayesian networks are directed graphical models, whereas Markov random fields are undirected models. It is also possible to have combinations of these. It is important for PPAML to support both kinds of models (although not necessarily in a single PPS). Parametric models have a fixed number of parameters, and hence, a fixed model structure. Many of the best-performing machine learning methods are non-parametric, which allows them to optimize predictive accuracy by adapting the complexity of the model to the complexity of the data. Developments in probabilistic modeling, such as Gaussian processes and Dirichlet process mixture models, make it possible to define non-parametric probabilistic models that express uncertainty in the model structure.

The third dimension concerns the structure of the objects over which inference is performed. Parametric models such as Hidden Markov Models can be applied to reason about variable-sized objects (such as protein sequences of varying length), and this is typically more complex than reasoning over fixed-sized objects. For many problems, the number of objects is not known—for example, when tracking multiple vehicles using noisy sensors or resolving co-referring expressions in natural language— and models must explicitly reason about an unknown number of objects.

The Query Structure: Once a probabilistic model has been fit to data, the primary computational task is to provide values for some of the variables in the model and make queries about other variables. We identified four important dimensions along which the query processing should vary in PPAML: Query Type, One Shot versus Tracking, Operational Tempo, and Stationary versus Change Points. A posterior query asks for the posterior joint distribution of the query variables, whereas a Maximum Aposteriori Probability (MAP) query asks for the values of the query variables that jointly maximize the posterior. In Posterior Marginal and MAP Marginal queries, some of the variables in the model are "free" in the sense that their values are not observed, and they are also not queried. Inference can be significantly harder in this case than when all variables are queried. For decision making, the query often involves computing the expected value of some variable, where the expectation is taken with respect to some (marginal) posterior. This can be more challenging because inference cannot necessarily ignore low probability cases if the value of the query variable is very large. Finally, in many cases of interest, the query seeks to find anomalous (i.e., low probability) data points. Again, this poses a challenge for some inference algorithms, because the focus is on the rare cases rather than the common ones.

A standard query is a one-shot query. One-shot queries do not consider the results of previous queries. But many applications involve making repeated queries as the input data evolves over time. We call this a tracking query. It is often possible to exploit the sequential nature of a tracked system to make these queries faster than if they are viewed as a series of one-shot queries.

Operational tempo characterizes the frequency of queries. The frequency can vary wildly between problems. For systems with rapidly-changing dynamics (e.g., quad-rotor aircraft), the tempo is extremely fast whereas other tracking problems occur at the scale of days or weeks, and hence exhibit a slow tempo.

Finally, in many applications, the assumption is made that the underlying phenomenon being modeled is not changing. Consequently, the data can be treated as an independent and identically distributed sample from the model. However, in other problems of interest, the distributions do change over time. In such cases, a change point detection query asks the system to identify when the change occurred (and to characterize that change).

3.1.1 CP1 – Unmanned Autonomous Systems (UAS) SLAM

Unmanned Autonomous Systems are playing an expanding role in achieving missions critical to national security, natural resource management, emergency response and emerging industrial applications. To achieve the goals of these various missions, fundamental advancements must be made that facilitate the development of sensing and control algorithms for these systems. At the heart of sensing and control of UAS are models that combine or "fuse" multiple sensors to predict the future state of the vehicle and the world it acts in. This challenge problem was based on two representative systems that were used to study the sensor fusion problem: a small automobile with a laser range finder, as well as a flying quad-rotor aircraft (a "quadcopter") that included a rich set of controls as well as visual, inertial, and location-based sensors. The challenge problem required PPS systems to take noisy sensor data and reconstruct both the path that the vehicle took through the world as well as a map of its surrounding environment.

Subject matter expertise for CP1was provided by Galois.

3.1.2 CP2 – Bird Migration

On peak nights during migration season, billions of birds take to the air across the US. However, because migration proceeds over vast temporal and spatial scales, and because it is difficult to observe directly, it is poorly understood. Scientists would like answers to questions such as (a) do birds wait for favorable winds before migrating (or are they on a fixed schedule)? (b) what factors influence a bird's decision to stop at some location? (c) what factors influence a bird's decision to resume migration? and (d) how do these factors vary from one species to another? Answering these questions requires constructing a model of the dynamics of bird migration.

In this challenge problem TA2-4 teams were tasked with:

- Near-term predictive accuracy (24 and 48 hours into the future)
- Reconstruction of bird population flows
- Coverage of confidence intervals or posterior credible intervals for the population flows and model parameters.

Subject matter expertise for CP2 was provided by Oregon State University.

3.1.3 CP3 – Automated Track Linking in Wide Area Motion Imagery (WAMI)

The challenge problem of automated track linking in WAMI originates in the automated video analytics domain, where target tracking algorithms produce short high-confidence tracks of moving objects, hereafter referred to as "tracklets". The descriptive and analytic value of these tracklets is greatly increased if they can be accurately stitched or linked together into longer

tracks following the same target, e.g., a moving vehicle, in the imagery. Tracking object movement from overhead imagery has both military and civilian applications. An improved solution to the linking problem could significantly advance the state-of-the-art in WAMI analyst exploitation tools. We used the publicly released Wright Patterson Air Force Base (WPAFB) 2009 dataset [1] from the Air Force Research Lab (AFRL). The Kitware tracker [2], assessed by AFRL as state-of-the-art, supplied moving object detections and high-confidence, short duration tracklets. Geographic context, such as roads, intersections, buildings, and neighborhood types were also provided, creating the framework for a rich, hierarchical probabilistic programming problem.

Subject Matter Expertise for CP3 was provided by Kitware.

3.1.4 CP4 – Small Problems Collection

The goal of the "Small Problems Collection" is to create a set of problems that span important dimensions of the space of probabilistic programs in terms of both program formulation and probabilistic inference. The set of problems was designed to help the PPAML TA2-4 teams identify important tradeoffs in the design and implementation of Probabilistic Programming Systems. Note that this is an important change in direction from the previous focus on benchmarking of PPS systems.

For the Small Problems Collection, we focused on dimensions that capture abstract problem structure. These include the following:
1. Data types: Continuous, Discrete, Mixed
2. Data structures: Atoms, Vectors, Relations, Grammars, Graphs
3. Model structure: Directed vs. Undirected, Parametric vs. Non-parametric, Fixed vs. Variable number of variables, Presence of latent variables
4. Query type: MAP, Marginal MAP, Expectations, Posterior Distribution, Posterior Summary (e.g., moments), Anomalies
5. Query timing: One-shot (data and query are presented together), Online (fixed query, but data arrive incrementally, so the query results need to be updated incrementally), Amortized (fixed data, but multiple, related queries arrive incrementally, and inference costs can be amortized across them), Online Amortized (both the data and the queries arrive incrementally).

Figure 2 places each of these small problems along these dimensions.

		Data Properties		Model Properties				Query				
	Problem	Types	Structure	Directed?	Parametric?	Fixed # Objects?	Latent variables?	MAP	Expectation	Posterior Distribution	Posterior Summary	Timing
1	Bayesian Linear Regression	Both	Vectors + Scalars	Directed	Parametric	Fixed	No	X		X	X	One-Shot
2	Medical Diagnosis (bipartite directed graph)	Discrete	Scalars	Directed	Parametric	Fixed	Yes	X	X		X	One-Shot
3	Discrete Time HMM	Both	Sequences	Directed	Parametric	Fixed	No	X			X	One-Shot and Online + Amortized
4	HDP + LDA	Discrete	Vectors	Directed	Nonparametric	Fixed	Yes	X		X	X	One-Shot
5	PCFG	Discrete	Sequences / Grammars	Directed	Parametric	Fixed	No			X		One-Shot
6	Network Analysis	Discrete	Undirected Graph	Directed	Parametric	Fixed	No	X		X		One-Shot
7	Friends and Smokers	Discrete	Relations	Undirected	Parametric	Fixed	Yes			X		One-Shot
8	Event Detection	Continuous	Time series	Directed	Parametric	Variable	No	X				One-Shot
9	Scalar Implicature	Discrete	Scalars	Directed	Parametric	Fixed	Yes			X		One-Shot
10	Lifted Inference	Discrete	Scalars	Directed	Parametric	Fixed	No	X				One-Shot

Figure 2 CP4 – Small Problem diversity characteristics

Subject matter expertise for CP4 was provided by Galois.

3.1.5 CP5 – Probabilistic Context-Free Grammars with Latent Annotation

Context-free grammars (CFGs) provide a simple model for the structure of language and are widely-applied in natural language processing systems. A CFG consists of a set of non-terminal symbols *N*, a set of terminal symbols *T*, and designated start symbol *S*, and a set of production rules of the form $N \rightarrow R$, where *R* is a sequence of terminals or non-terminals. For each nonterminal $n \in N$, the set of rules having *n* on their left-hand-side are the "rules for n", which we denote as *Rules(n)*. A sentence is generated by beginning with the start symbol *S*, choosing one of the rewrite rules in *Rules(S)* and replacing *S* by the right-hand-side of the chosen rule. This is repeated, each time expanding one of the non-terminals in the emerging sentence until no non-terminals remain.

Probabilistic context-free grammars (PCFGs) extend CFGs to define a probability distribution over the sentences and parse-trees generated by the grammar by specifying a separate multinomial distribution for each non-terminal *n* over *Rules(n)*. These probability distributions can be learned from data consisting of sentences and their parse trees ("treebanks"). The most famous treebank is the Wall Street Journal (WSJ) treebank developed at the University of Pennsylvania. It consists of 23 "sections", and it is available at no cost from the Linguistic Data Consortium (LDC) at the University of Pennsylvania.

A weakness of both CFGs and PCFGs is that each time a non-terminal is expanded using one of its rules, the choice is made independently of all other choices. This independence prevents PCFGs from capturing many important linguistic regularities. One way to address this problem is to replace very general non-terminals such as noun phrase (NP) and noun (N) with an expanded set of symbols. For example, we could have NP-animate, NP-inanimate, N-animate, and N-

inanimate, to separately model noun phrases and nouns referring to animate versus inanimate objects. These are sometimes referred to as annotations because one can imagine manually annotating each N and NP in a treebank with this additional information. However, there are no large annotated treebanks, and it is also not clear what annotations should be introduced.

One way to avoid manual annotation is to take the extreme approach known as lexicalization in which a separate "annotation" is defined for each word in the lexicon. For example, we would have N-car, N-truck, N-bus and so on. This leads to an immense grammar, and learning the parameters for the probability distributions requires introducing some form of "smoothing" so that rules for similar words are given similar probabilities.

In this challenge problem, we investigated a different approach pioneered by Matsuzaki et al. [3] in which the annotations are "latent". That is, we can view the parse trees in the treebank as having "missing" annotations. We specify that each non-terminal can have up to *k* annotations, and it is the job of the learning algorithm to determine what those annotations should be and which sentences should use which annotations. In statistical modeling terms, we replace each nonterminal by a mixture of distributions over its child non-terminals (which are in turn, recursively, mixtures over their children). Recent work on this model includes Petrov et al. [4] and Cohen et al. [5].

This problem was given in two phases.

Phase 1: Fitting a standard PCFG to a subset of the WSJ corpus.

Phase 2: Fitting a latent PCFG to a subset of the WSJ corpus. The number of latent annotations was fixed.

Phase 2 Hierarchical Dirichlet Process (HDP) option: Fitting a latent PCFG to a subset of the WSJ corpus. The number of latent annotations were flexible and modeled via a Hierarchical Dirichlet Process.

Training data consisted of a text corpus (from the WSJ section of OntoNotes 5.0) with associated constituency-based parse trees (i.e., one terminal associated with each nonterminal leaf node). The trees were binarized, which is a standard transformation intended to simplify the parsing code.

Subject matter expertise for CP5 was provided by Galois and Oregon State University.

3.1.6 CP6 – Image Labeling

The problem of multimedia retrieval is to develop the scientific methodology to understand and discover images/videos with particular content from a complex, large, and growing collection of multimedia. Real-world multimedia, especially as shared on the Internet, can be challenging to retrieve using only visual information, due to complex content, partial occlusion, and diverse styles and quality. The most common solution to this problem is to annotate media with keywords that describe the content and then perform a keyword search against these annotations.

The problem of annotating images consists of inferring content labels, *L*, conditioned on an image, *I*, and other related metadata information, M, e.g., $P(L|I,M)$.

In Challenge Problem 6, we approached the task of automatic image annotation or labeling by exploiting the metadata, *M*, in addition to the visual information, *I*. Some types of metadata (i.e., Exchangeable Image File Format (EXIF) tags) are generated by the camera when the image is taken; others (i.e., user-provided tags, comments from viewers) are generated after the image is uploaded to an image-sharing service such as Flickr. We used a subset of the MIRFLICKR [Huiskes2008] dataset to supply the ground-truth image labels, image features, and related metadata.

Within the PPAML taxonomy of challenge problems, this CP is related to the Intelligence Analysis domain; the data structures are a hybrid of discrete (categorical) and continuous (features and feature distances) presented in both relational and vector forms. The basic parametric probabilistic model is an undirected graphical model over a fixed model structure with latent variables. Queries are formulated as marginal *maximum a posteriori* MAP for individual images, or joint MAP for the entire graph. The query timing is one-shot with slow tempo and stationary parameters.

Subject matter expertise for CP6 was provided by Kitware.

3.1.7 CP7 – Flu Spread

Predicting the spread of epidemics through space and time can help government agencies and organizations better prepare and allocate resources. Seasonal flu epidemics have been closely monitored and many years of historical data have been collected by the medical community. The data collected and aggregated by Centers for Disease Control and Prevention (CDC) has been valuable for researchers trying to develop models to forecast the spread of flu epidemics. In addition to the CDC data, there are many other data collected by different entities for various purposes – many of them unrelated to flu epidemics. When those datasets are considered together with the CDC data, they offer the opportunity to significantly improve our ability to assess and forecast flu epidemics both spatially and temporally. Datasets include social network data and vaccination statistics. Those data have different characteristics (e.g., percentages for CDC regional Influenza-like Illness (ILI) rates and flu vaccination, and quantized flu activity levels for CDC state ILI rates) and different spatial and temporal resolution. Aggregating the data into a forecasting model is challenging, but if successful, can provide much-improved forecasting accuracy over a longer time horizon than what current approaches based on limited sources of information can accomplish.

The problem was given in three phases.

Phase 1 Problem (Reconstruction)
During Phase 1, the goal was to fuse multiple data sources to reconstruct Influenza-like Illness rates at a spatial resolution finer than that of the ILI data from CDC. Performers were to estimate weekly ILI rates in the 48 contiguous states. The spatial resolution of the estimates was at the county level. The results were compared to a set of "Evaluation Regions" consisting of state-

level ILI rates from selected states (Massachusetts, North Carolina, Rhode Island and Texas) and district-level ILI rates from 2 states (Mississippi and Tennessee), where each district consists of multiple counties.

Phase 2 Problem (Nowcasting)
During Phase 2, the goal was to produce estimated ILI rates that are timelier than those published by the CDC (while maintaining spatial resolution finer than the CDC). The ILI data from CDC and the Evaluation Region states are released after a delay of 1 to 2 weeks. The goal of Phase 2 was to predict ILI rates in week *t* using all data from previous weeks *t* - 1, *t* - 2, …,t - n. This will include the CDC ILI rates from week *t* - 2 and the Twitter data from week *t* - 1.

Phase 3 Problem (Nowcasting continued)
The task in this phase of CP7 was to predict seasonal rates of Influenza-Like Illness in 60 distinct sub-populations of the continental US, ranging in size from the entire country to individual counties.

In addition to historical ILI rate data for each population, three different kinds of covariate data, representing flu-related tweets, vaccination claims, and weather, were provided for use in solutions. In a *simulated forecast* experiment, all four kinds of variables were made available to solutions, one week of data at a time, over the 32-week target season.

Subject matter expertise for CP7 was provided by Scientific Systems.

3.1.8 CP8 – Desktop Activity Recognition

Desktop knowledge workers perform a wide variety of procedures or activities each day as they carry out their work. Much of this work is repetitive, so there has long been interest in providing intelligent assistance for these tasks. One such effort was the Activity Recognition and Proactive Assistance (ARPA) effort within the Cognitive Assistant that Learns and Organizes (CALO) project led by SRI International in 2006-7, which was the inspiration for this challenge problem.

We modeled each user as having a library of parameterized workflow procedures. For example, one procedure would be to provide edits on a document and return it to the primary author. This procedure consists of the following finite-state machine:

1. Receive incoming EMAIL_IN with an attachment FILE_IN
2. Save the attachment as FILE1
3. Open FILE1 in Word
4. Repeat some number of times:
    a. Make edits to FILE1
    b. Save FILE1
5. Create EMAIL_OUT as a reply to EMAIL_IN
6. Attach FILE1 to EMAIL_OUT (the attached file is known as FILE_OUT)
7. Send EMAIL_OUT

Of course, there could be many variants of this procedure. For example, in Step 2, the user might Open the attachment and then use a SaveAs to create FILE1. In step 4b, the user might invoke

SaveAs to create a new file FILE2, which would then be attached to the email message in step 6. Finally, the user might close FILE1, exit Word, and then later reopen FILE1 in Word.

One way that an intelligent desktop assistant could help the user would be to learn a set of workflow definitions, detect the start of each new workflow instance, and track the state of each workflow. Using this information, the assistant could offer to automatically execute some of the workflow steps via an automation interface. For example, when the user performs step 5, the assistant could offer to attach FILE1 to the message.

Desktop knowledge workers are famous for multitasking. It is extremely common for the user to start a workflow and then be interrupted by many other workflows before resuming work on the original workflow. An assistant could also maintain a list of the active workflows and make it easy for the user to select a workflow and resume work on it (e.g., by restoring the state in step 3).

In this challenge problem, teams were given a sequence of events recorded from human subjects engaged in desktop work. These workflows were interleaved, with multiple instances of each workflow active simultaneously, but with different parameters. The inference task was to correctly assign each observed event to the correctly-parameterized workflow instance.

In Phase 1, teams processed the events in "batch mode" to learn the workflow. Then, given a new sequence of events, their system should produce a MAP assignment of workflow identifiers (with parameters) to each event in the sequence. In Phase 2, teams processed the test sequences incrementally and computed a posterior distribution over workflow ids (and parameters) immediately after each event is observed. The metrics in Phase 1 measure accuracy, while the metrics in Phase 2 we also considered timeliness.

Subject matter expertise for CP8 was provided by Oregon State University.

3.1.9 CP9 – Hackathon I: Data Analysis

Beginning with Challenge Problem 9, the TA1 team in collaboration with DARPA restructured the approach and executed evaluations as Hackathon events as part of the program PI meetings.

The goal of the first Hackathon was to demonstrate the capabilities of probabilistic programming for a variety of tasks that arise in data analysis. The Hackathon was not structured as a competition. Instead, our intent was for performer teams to collaborate in order to best achieve the goal of demonstrating the capabilities of probabilistic programming.

We developed a data set based on the Gapminder database. This database consists of data on 519 variables by country and by year. We chose approximately 80 parallel time series that were of interest and had reasonable coverage and included many missing values. Some of the variables in the time series can be viewed as "intervention" variables and other variables can be viewed as "outcome" variables, but many variables could be viewed in multiple ways and may constitute important intermediates.

TA2-4 teams were given the following tasks to complete during the two-day hackathon event:

Task 1: Impute missing values. Create a joint model of the data and then sample from the joint posterior over the missing values.

Task 2: Generate causal hypotheses about the effects of intervention variables on outcome variables. Fit models of the outcome variables as a function of intervention variables (and potential confounding variables as appropriate) to generate causal hypotheses about the effects of intervention variables on outcome variables.

Task 3: Model the processes that are causing missing values. Determine cases where the missing values are not missing at random. Build models of the missingness processes and incorporate those models into the imputation process of Task 1 if possible.

Task 4: Criticize and refine a given model. Identify weaknesses in the TA1-provided model. Develop a refinement of the model (or completely replace it) to obtain a better model.

Task 5: Compute optimal decisions. Using the model(s) from Task 2 and given a fixed budget, determine the optimal allocation of the budget in order to benefit the largest number of people.

Subject matter expertise for CP9 was provided by Oregon State University.

3.1.10 CP10 – Hackathon II: Weather Data

The goal of the final PPAML hackathon was to demonstrate the power of probabilistic programming to support *extrapolative generalization*. Most statistical learning methods only perform *interpolative generalization*—that is, they interpolate between the given training examples to answer new queries. A consequence of this is that to work well across a wide variety of queries; they must be given very large training sets that provide examples of the full range of variability of the problem. In contrast, probabilistic programming methods have the potential to represent background knowledge and combine it with the training data to extrapolate beyond the training data.

The data for this exercise is provided by the Oklahoma Mesonet (OK Mesonet). In an NSF-funded project, Oregon State University collaborated with OK Mesonet on new methods for automated data quality control. Weather network sensors frequently break or drift out of calibration. The goal of automated data quality control is to detect broken and uncalibrated sensors and flag them for visits by technicians. Similar problems arise in other sensor networks and emerging Internet of Things applications.

The OK Mesonet consists of approximately 130 weather stations distributed across the state (see Figure 3). Stations in southeast Oklahoma are located at elevations in the 100-300m range above sea level and are heavily influenced by the Gulf of Mexico. Stations in northwest Oklahoma are at elevations in the 500-1350m range and are more influenced by winds originating in front of the Rocky Mountains.

The dataset provided by the TA-1 team consisted of a set of stations that lie along a transect extending from southeast Oklahoma to northwest Oklahoma. The stations along this transect are divided into three groups. Six stations in the middle of the transect constituted the training-only stations. Data from four southeastern-most stations was provided for both training and testing. And data from four northwestern-most stations formed the test set. To successfully answer the queries, the teams were required to extrapolate in space and time.

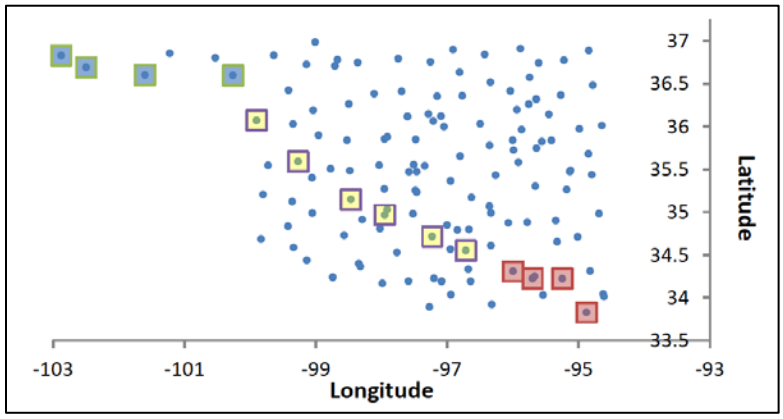

Figure 3 Locations of Oklahoma Mesonet weather stations. Stations along the transect are highlighted.

TA2-4 teams were given the following tasks to complete during the two-day hackathon event:

Task 1: Coarse-Scale Prediction. The goal of this task is to predict summary statistics including monthly mean, high, and low temperature, mean, max, and min relative humidity, mean, max, and min solar radiation, and mean, max, and min wind speed. This query will be answered without any data from the test region stations.

Task 2: Pure Prediction. In the pure prediction query, the teams will predict the readings for temperature, pressure, relative humidity, and solar radiation at a station without any other information about that station except its location and elevation.

Task 3: Conditional Prediction. In the conditional prediction query, the teams will be given data at a station for some sensors (e.g., pressure, solar radiation, wind speed, and direction) and be asked to predict the values of other sensors (e.g., temperature and relative humidity).

Task 4: Imputation. For this query, the teams will be given data for all sensors at one of the northwestern stations during the test period, but a substantial fraction (e.g., 40%) of the observations will be missing. The task will be to impute the missing values. This will require various conditional prediction models.

Task 5: Quality Control. For this query, the teams will be given data for a station. The data will contain inserted sensor failures based on a sensor failure simulator that we have developed. The teams will be asked to identify which sensor readings are the result of

broken or miscalibrated sensors. This is very similar to the Imputation task, but it tests how well the teams can estimate the uncertainty of the imputed/predicted values.

Subject matter expertise for CP10 was provided by Oregon State University.

3.1.11 Challenge Problem Artifacts and Future Research

Over the course of the PPAML program, Galois facilitated the distribution of challenge problem artifacts to many non-affiliated research initiatives. With the end of the program performance period, Galois has archived all challenge problem artifacts and made them publicly available.

3.2 Evaluate Probabilistic Programming Systems

Galois was responsible for evaluating the performance of each PPS on each challenge problem, measuring both the quality of the solutions and the run-time performance. A report summarizing each TA2-4 PPS results on each challenge problem was written and made available to all program participants, as well as presented at each PI meeting.

3.2.1 Procedures for Evaluating Challenge Problems

Challenge problem evaluations were structured in six-month cycles. The general structure of each evaluation cycle consisted of:

- Challenge Problem Introduction – At each PI meeting a subset of Challenge problems (including sub-phases) were introduced to the TA2-4 performers. The introductions consisted of a problem description, evaluation metrics, and evaluation milestone schedule.

- Beta Evaluation Phase – Typically at the start of month three of the evaluation cycle, Galois opened a beta evaluation phase where we would accept PPS solution submissions, run evaluations and provide evaluation results. This was an iterative process, supporting PPS solution refinement.

- Final Evaluation – Two weeks before the PI meeting, the beta submission window closed. Galois ran final evaluation for each PPS generating a results report for DARPA and TA2-4 teams.

- Reporting of Results – Results from the evaluation cycle were presented at the PI meeting and distributed to TA2-4 teams.

3.2.2 Quantitative Evaluation

Individual challenge problems explicitly defined the criteria for measuring the quality of PPS solutions as part of the problem description. In addition, Galois collaborated with TA2-4 teams to solicit feedback. With each PI meeting, after problem introduction, Galois hosted breakout sessions where teams could provide feedback and help shape problem evaluation criteria.

3.2.3 Qualitative Evaluation

In assessing results of the first two evaluation cycles in Phase I of the program, Galois identified an opportunity to leverage our deep experience developing high-level and domain-specific languages. While the quantitative metrics were critical to the development of PPSs, this fell short in assessing and providing valuable feedback on usability. Galois, having more than twenty years of experience developing high-level and domain-specific languages had the resources available to fill this need.

In evaluation cycle three Galois introduced a pedagogy experiment as part of the evaluation criteria with TA2-4 teams. The goals of the experiment were to:

- Improve PPS features and usability
- Expose gaps in documentation
- Improve communication with/across teams

A series of pair programming sessions between individual TA2-4 teams and Galois were run where we implemented and explored a series of simple models. This enabled Galois to work directly in the PPS, facilitating deeper knowledge of PPS capabilities, usability and language design. Feedback was provided directly to TA2-4 teams as well as summarized as part of the evaluation results at the PI meetings.

3.2.4 Evaluation Infrastructure

A priority for Galois as evaluator was to be able to support many performers wishing to rapidly iterate on challenge problem solution development. It was critical that for any PPS submission during the beta evaluation phase of a cycle, Galois have the ability to run the evaluation and provide results quickly to enable performers to refine their solutions.

Galois developed an evaluation framework to facilitate PPS evaluation runs. An overview of the evaluation framework is shown in Figure 4. A specification defining PPS submission requirements was developed and shared with TA2-4 teams. This specification defined PPS packaging requirements, installation requirements and execution requirements. PPSs adhering to these requirements enabled Galois to drop in new solutions and automate the evaluation execution.

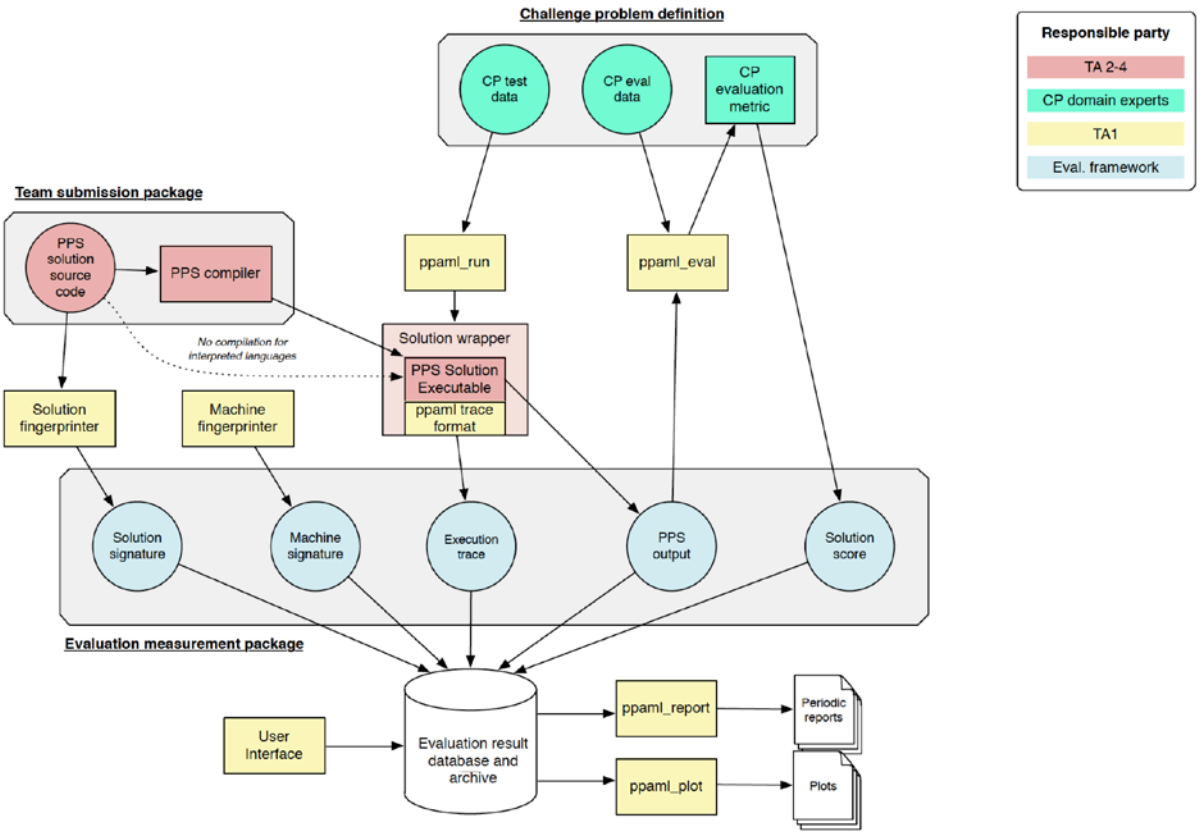

Figure 4 Evaluation Framework

3.2.5 Team Challenge Problems

Goals and Impact

One objective of the PPAML program was to develop user scenarios in a variety of application domains where PPS tools can be applied.

As part of this objective, each TA2-4 team was responsible for defining a Team Challenge Problem (TCP) to demonstrate and evaluate the capabilities of their PPS tools. By year two of the program, this requirement had not yet been met. The TA1 team proposed a process for satisfying the Team Challenge Problem requirement that leveraged the expertise gained over the first two years managing TA2-4 challenge problems. This was agreed to by the Government and executed over the final two years of the program.

The goal of the Team Challenge Problems was to demonstrate the merits of probabilistic programming to a broad scientific and technical audience. The ideal TCP exploits probabilistic programming to demonstrate novel capabilities on a compelling domain problem. This could include advancing the state of the art in some established problem or achieving interesting performance on a novel problem.

A total of six TA2-4 and TA3 teams submitted a Team Challenge Problem for evaluation: Charles River Analytics, Gamalon, Indiana University, Massachusetts Institute of Technology, Applied Communication Sciences, and SRI International.

TCP Management and Evaluation Plan

Satisfying the Team Challenge Problem requirement divided naturally into two phases: Defining the problems and evaluating the solutions. The tasks for each phase are described in the following sections.

Phase 1: Defining Team Challenge Problems

The TA1 team developed a request for proposal (RFP) detailing the TCP requirements and used to measure acceptance of performer proposals. The requirements covered the following dimensions: problem description and justification, feasibility, evaluation metrics and approach, dissemination, and suggested reviewers.

The TA-1 team reviewed submitted proposals against the following criteria:

a. Relevance to DoD and the goals of the PPAML program.
b. Potential to solve a novel problem or beat the state of the art on an existing problem.
c. Extent to which probabilistic programming will be the key enabling factor for success.
d. Degree to which the metrics are appropriate to the problem domain.
e. Extent to which the evaluation protocol supports an independent test.
f. Likelihood that the challenge problem data and evaluation can be publicly replicated.
g. Feasibility of the engineering plan for evaluation.

The Government reviewed the analysis before sharing the feedback with each submitting team as well as final proposal approval.

Phase 2: Evaluating the Team Challenge Problems

Team challenge problem submissions were evaluated in two rounds. The TA-1 team evaluated submitted solutions using the evaluation metrics identified in the proposals. Evaluation results were provided to each team to include in their PI meeting status presentations. The TA-1 team met with each team prior to the PI meetings to arrive at an appropriate interpretation of the results.

TA-1 Team Challenge Problem

Building on our experience working with other performers in a TA1 role, and also leveraging the general programming languages expertise at Galois, we developed a probabilistic programming language (PPL) called Grappa. The goal of Grappa was to develop an approach to PPLs that could utilize special-purpose, efficient inference methods while still providing the expressivity and productivity benefits of existing general-purpose PPLs. The tension between these two goals — efficiency versus expressivity – manifests itself in terms of the *representation* used for the programs written in a PPL. A more general representation can represent more programs, yielding

a language with a higher expressivity, but efficient inference algorithms, such as Hamiltonian Monte-Carlo (HMC) sampling or Belief Propagation (BP), require more specific information, leading to special-purpose representations for each algorithm. Further, many of these special-purpose representations rule out whole classes of probabilistic programs; e.g., HMC requires programs to be represented as probability functions with gradients, which allows only continuous, real-valued variables, while BP requires programs to be represented as Bayesian networks, which allows only discrete variables. These representations are not only in direct conflict with our expressivity goal, but can often, as in HMC and BP, be in conflict with each other.

To overcome these difficulties, we have designed Grappa to be a *multi-targetable* PPL compiler, that can target a wide variety of different representations, as depicted in Figure 5. As input language, Grappa provides a high-level, expressive, functional language similar to many other PPLs, providing a high level of expressivity. For the subset of Grappa programs that can be compiled to a particular representation – for instance, programs with only continuous, real-valued variables, which can be compiled to probability functions with gradients – Grappa can compile these programs to that representation, allowing special-purpose, efficient inference methods to be applied to that program. For those programs that fall outside of the subset, Grappa will emit compiler errors that inform the user of the mismatch. In this way, Grappa can support a wide variety of efficient inference methods in the same high-level, expressive language.

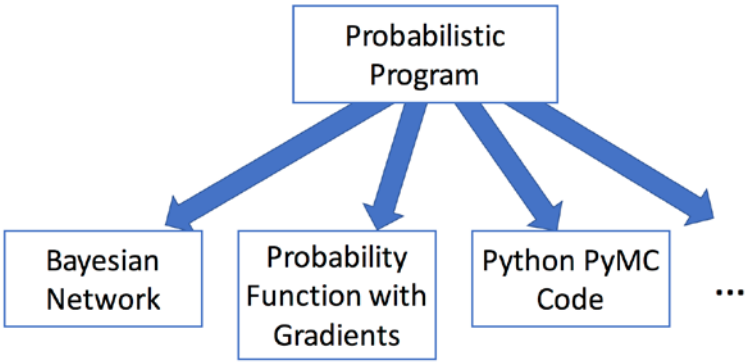

Figure 5 Grappa- a multi-targetable PPL compiler

Grappa currently includes support for a number of useful representations, such as probability functions with gradients and Bayesian networks, allowing a number of existing, efficient inference methods, such as HMC and BP, to be applied. Also included is a representation for compiling to PyMC models, a commonly-used Python package for performing Markov chain Monte Carlo (MCMC) sampling, thereby allowing Grappa to leverage the inference methods provided by that package. Additionally, Grappa provides a plugin architecture, that allows the set of representations it supports to be extended. This in turn opens up new opportunities for experimenting with new inference approaches, allowing researchers to try out new representations and/or combine existing representations without having to write a whole new compiler. For instance, we have already begun investigating a number of new ideas using this technology, including:

- Interleaved HMC sampling, for real-valued variables, and BP-based sampling, for discrete variables, an approach that is particularly amenable to sampling, for instance, the parameters of hidden Markov models;

- Parallelized sampling of independent variables, to enable speedups for models that can be decomposed into independent groups of variables;

- Representations for Dirichlet process mixture models, a powerful class of models that have recently been finding widespread applicability in a number of domains.

A key difficulty in supporting this wide array of compiler targets is that compilers are complex, difficult to write, and error prone. To avoid this problem, the Grappa plugin architecture for supporting new representations allows users to define representations using *abstract interpretation*. Abstract interpretation is an advanced programming languages technique for performing program analyses and transformations in a way that is *compositional*, meaning that the analysis or transformation of a whole program is built up from that of the pieces by combining those pieces in a straightforward, regular way. Rather than writing an entire compiler, a Grappa plugin writer need only provide an *interpretation* for the constructs of the Grappa language that a particular plugin supports, as Haskell type class instances. The Grappa compiler then combines the interpretations of the individual constructs of a program into an interpretation of the whole program, and also takes care of concerns that are universal across all interpretations, such as parsing, code generation, etc. Interpretations are much simpler to write and to debug than compilers because, rather than writing a translator from one syntactic representation to another, like a compiler, the interpretation of a Grappa construct is itself just code in the implementation language, or meta-language, used to write Grappa itself, which in this case is Haskell. That is, we can inspect, unit test, debug, etc., each piece of an interpretation just like standard Haskell code, instead of having to consider all possible ways a compiler could generate code. Further, interpretations are also type-checked, just like standard Haskell code, which also prevents a wide class of errors in the compiler that could generate ill-formed code.

3.3 Annual Summer Schools

Galois developed and led four annual summer school sessions. The objectives of the summer school as part of the PPAML program were to:

- Grow a research community interested in probabilistic programming,
- Create a community of practitioners with experience in probabilistic programming,
- Identify potential transition candidates and help them assess the suitability of probabilistic programming for their problems, and
- Provide systemic feedback to the PPS development teams regarding the usability of their probabilistic programming systems.

In pursuit of these objectives, each summer school was designed to teach participants the foundational background required for probabilistic programming, give them an opportunity to work directly with the creators of probabilistic programming languages being developed as part of PPAML, and apply the tools to specific domain problems.

Online surveys and personal interviews were utilized during the summer school to elicit feedback from the participants about the lectures, the PPSs as well as the content and structure of the summer school itself. Feedback was shared with both DARPA and participating TA2-4 teams.

3.3.1 Structure of the Summer School

Each summer school session was structured as a two-week session. Utilizing participant and presenter feedback, along with our own observations and experience, Galois evolved the structure and format over the program. By year three, and continued in year four we found the ideal structure and approach to include:

- One TA2-4 team presenting their PPS solutions each week
- A mix of foundational content, lecture and hands-on exercises
- Provide time for participants to work on their real-world projects, with results and experience presented at the end of each week

Figure 5 shows an example outline from the 2016 session.

	Monday - 7/25	Tuesday - 7/26	Wednesday - 7/27	Thursday - 7/28	Friday - 7/29
9:00	Introduction to Summer School	Lecture: Introduction to Functional Programming and Clojure	Lecture: Introduction to Anglican	Lecture: Advanced Inference in Probabilistic Programming	Project Free Coding
10:00	Overview of PPAML Program	Functional Programming Excercises	Anglican Programming Excercises	Project Free Coding	
11:00	Foundations Presentation				
12:00	Lunch	Lunch	Lunch	Lunch	Lunch
13:00					
14:00	Lecture: Introduction to Probabilistic Programming	Lecture: Introduction to Probabilistic and Generative Modeling	Anglican Programming Excercises	Lecture: Contributing to Anglican	Participant Presentations
15:00		Lecture: Introduction to Inference		Project Free Coding	
16:00		Probabilistic and Generative Modeling Excercises	Project Brainstorming		

	Monday - 8/1	Tuesday - 8/2	Wednesday - 8/3	Thursday - 8/4	Friday - 8/5
9:00	WebPPL: Introduction	Agents	Approximate Inference Algorithms	Data - Analysis and Prediction	Project Presentations
10:00					Wrap-up
11:00					
12:00	Lunch	Lunch	Lunch	Lunch	Summer School Closes
13:00					
14:00	Q&A with Noah Goodman	Agents Continued	Inference Algorithms Continued	Analysis and Prediction Continued	
15:00		Project Free Coding	Project Free Coding	Project Free Coding	
16:00					

Figure 6 Structure of the 2016 Summer School

3.3.2 Summer School Participants

Participants at each summer school included students, researchers, and practitioners from industry who work on problems that may benefit from probabilistic programming. Collectively, the participants were well versed in Mathematics and Computer Science and had hands-on experience with multiple programming languages. Generally, building probabilistic models, thinking generatively, and fitting data stood out as areas where participants lacked similar depth.

Appendix A contains more detailed demographic information about the participants and compares it to that of the participants from each of the four summer school sessions.

3.3.3 Summer School Presenters

Over the course of the program nearly all TA2-4 teams as well as two TA-3 teams taught and presented their PPSs at one or more Summer School Sessions. Typically presenting teams brought five or more members to ensuring plenty of support for participants. The presenting teams, and PPSs featured over the course of the program:

- 2014 Summer School
  - Massachusetts Institute of Technology (MIT); Venture
  - Charles River Analytics (CRA); Figaro
  - University of California; Berkeley; BLOG
  - Stanford; Church
  - Gamalon; Dimple/Chimple
  - Indiana; Harkaru
- 2015 Summer School
  - CRA; Figaro
  - MIT; Venture
- 2016 Summer School
  - Invrea; Anglican
  - Stanford; webPPL
- 2017 Summer School
  - MIT; BayesDB, CrossCat, Venture, Gen
  - Gamalon; Particle, Tycho

3.3.4 Feedback Methodology

The feedback elicited during the summer school was intended to provide insights about user experiences, perceptions about the probabilistic programming tools, and broader observations to enhance the PPAML program and associated summer schools.

Participants completed questionnaires for each of the major phases of the summer school: On the first day of summer school, after the foundations lecture, after each of the initial language lectures, end of each language session, and on the last day of summer school.

Raw questionnaire results were immediately shared with presenting teams. Results were also used in each session report where major themes were summarized.

3.4 Program Collaboration

Galois configured, deployed and maintained technical infrastructure for supporting PPAML project collaboration. The primary mechanisms and infrastructure included:

- Wiki Servers: one public facing server, and one team-only server accessible by the Government as well as all TA1-4 teams. This proved an effective mechanism to maintain and communicate program information (schedules, contact information, challenge problem information hub's). In addition, the wiki's were used to support Summer School sessions as a central repository for participants to access session information, agenda's, and presented materials.

Periodically the TA1 team was contacted by researchers or people in industry not affiliated with the program. The wiki proved an effective source to connect their interests with PPAML materials.

- A set of mailing lists for PI communication as well as to support each challenge problem.

- A Mattermost server was deployed providing enterprise class team collaboration capabilities for Summer School collaboration.

- Multimedia Digital Archiving System (MIDAS) data distribution server to support data hosting for the challenge problems. For challenge problems one through seven, Galois distributed all challenge problem data to TA2-4 teams via the MIDAS infrastructure.

- OwnCloud, a cloud based file sharing tool was used in challenge problems one through seven for solution staging. TA2-4 teams submitted all submissions via the ownCloud portal.

- For challenge problems eight through ten, Galois hosted a Gitlab instance to support all aspects of challenge problem development and evaluation with TA2-4 teams.

- Galois has held regular phone calls to facilitate collaboration. These include: bi-weekly calls with the Government; weekly calls with the TA1 team participants; as well as frequent albeit not regularly scheduled calls with TA2-4 teams.

- The pedagogy sessions with the PPS design teams created closer working relationships and proved valuable in the clarification of assumptions and elaboration of questions.

- A Google-hosted discussion forum was created to further cross-TA team communications and help identify significant technical topics of importance to the PPAML program. The forum was community-driven and moderated by Galois.

- A Google-hosted repository was created to hold PPAML-related assets created by the various TA teams. Assets included in the repository were staged with the goal to transfer the collected assets to the DARPA-hosted public repository at the end of the PPAML program.

4.0 RESULTS AND DISCUSSIONS

Galois, as the TA-1 research evaluator, successfully achieved our program objectives.

Develop Challenge Problems – Galois delivered ten challenge problems spanning the set of metrics laid out at the program kickoff. A new challenge problem was fully defined and introduced every six months at each program PI meeting, with three problems introduced at the program kickoff.

Galois worked closely with the Government and TA2-4 teams reviewing the results of each evaluation cycle, identifying opportunities for future problems. A significant result of this

continuous improvement effort was the restructuring of the final two challenge problems as Hackathon events. With the culmination of the four-year program, these events were designed to test and evaluate PPSs against the original program objectives and the maturity of the performer PPSs.

All challenge problems were designed, built and prepared with the intent to be made publicly available for future research efforts. Artifacts for all challenge problems are publicly available. This includes the problem descriptions, datasets and detailed documentation.

Evaluate Probabilistic Programming Systems – To support a large set of PPS evaluations multiplied by an iterative evaluation approach, Galois developed an evaluation architecture, PEVAL, which automated the evaluation process. A specification defining PPS submission requirements was developed and served as a contract with TA2-4 teams.

Galois emphasized communication and collaboration with TA2-4 teams, which contributed to the successful completion of solution evaluations. This included kick-off meetings to review and answer questions regarding the challenge problems, on-going check-ins to answer questions and support solution development as well as providing evaluation results along with our assessment and insights within twenty-four hours of the completion of any evaluation run.

Evaluation results were successfully completed every cycle with results shared with TA2-4 teams and presented at the corresponding PI meeting. At each PI meeting Galois hosted break-out sessions which as part of the agenda reviewed the evaluation results and working with attendees identifying opportunities for future evaluation cycles.

With the introduction of Team Challenge Problems in the second half of the program, Galois performed independent evaluations across two phases for each of the submitted solutions. Galois collaborated closely with the submitting teams to setup the proper evaluation environment. With each submission, we ran the experiments and provided written results within one week. The results included the metrics achieved as well as a qualitative assessment and inputs. Galois also supported TA2-4 teams in the development of the results they presented at PI meetings.

Organize and Run Summer Schools – Galois successfully organized, planned and facilitated four summer schools over the course of the program. Achievements as measured against the program goals defined for this activity:

- **Grow a research community interested in probabilistic programming. Create a community of practitioners with experience in probabilistic programming**. This objective was successfully achieved. There was a total of 105 participants with 52 coming from Academia and 53 from Industry. Several companies had additional participants attend in subsequent sessions to continue to develop expertise in probabilistic programming in their organizations.

- **Identify potential transition candidates and help them assess the suitability of probabilistic programming for their problems.** Integrated into the summer school curriculum was the ability for participants to bring their own domain problems and work

on them during the session. Participants then presented results they achieved, as well as an assessment of the tools and approaches for their particular problem. As the PPSs matured through the program, the final two summer school sessions generated several collaboration and transition opportunities between participants in industry and PPS teams.

- **Provide systemic feedback to the PPS development teams regarding the usability of their probabilistic programming systems.** Feedback from participants was collected through surveys, direct conversation and participant project result presentations. The raw survey results and project presentations were shared with the PPS development teams. In addition, a report summarizing the feedback, and major themes was produced after each summer school and shared with both the Government and PPS development teams. In addition to the systematic feedback described, each PPS development team took advantage of the opportunity to work directly with participants and get direct feedback throughout each session.

Foster Collaboration – An emphasis of the TA-1 team was a continual focus on enabling collaboration amongst program performers. Galois setup and hosted several collaboration tools, data repositories and communication forums to support collaboration workflows. Over the course of the four-year program, Galois continued to evolve the workflows and tools. A continuous improvement effort was practiced to take advantage of new collaboration technology as well as identify new workflows that could have positive impact on the program goals.

Grappa – In the final Phase of the program, Galois was able to develop Grappa, a PPL, which is currently being applied on two government-funded programs, for the purpose of detecting cyber-attacks. The first is the Air Force Research Laboratory's Malware Detection program. For this program, we are using Grappa to detect cyber-attacks on UAVs, by training a collection of hidden Markov models to detect communication patterns between processes on a UAV that differ from "normal" communication patterns. The second is DARPA's Transparent Computing program, wherein we are applying Grappa to detect advanced persistent threats in enterprise networks. For this program, we are applying Grappa to learn "normal" patterns of process forking, to detect, e.g., Firefox processes forking unexpected system processes that could indicate a cyber-attack. Grappa is also currently in the process of being open-sourced and moved to GitHub. Finally, as part of the Grappa project, we have also made theoretical advances in PPL semantics, by defining a denotational semantics based on Lebesgue integration that also supports higher-order functional programs. This combination of semantic features, which has so far been elusive to semantics researchers, is the subject of a submission to the International Conference on Programming Languages Design and Implementation 2018.

5.0 CONCLUSIONS

As the TA-1 Program Evaluator, we successfully met our primary program objectives.

1. **Develop Challenge Problems** – A total of ten challenge problems were delivered spanning a broad range of characteristics designed to ensure that over the course of the program the population of challenge problems covered a wide range of domains and problem types.

2. **Evaluate Probabilistic Programming Systems** – The TA-1 team completed all evaluations on schedule, measuring the performance of PPSs on each challenge problem with results presented at each PI meeting.

3. **Organize and Run Summer Schools** – Over four summer school sessions, a total of 105 participants from both Academia and Industry attended the program. The summer school track of the PPAML program was an effective approach to growing the research community interested in probabilistic programming, developing a community of practitioners with experience in probabilistic programming, identifying transition candidates and opportunities and providing systematic feedback to the PPS development teams regarding the usability of their probabilistic programming systems.

6.0 REFERENCES

1. [Dataset] Air Force Research Labs. SDMS: WPAFB 2009 Dataset. https://www.sdms.afrl.af.mil/index.php?collection=wpafb2009, 2009.

2. [Paper] Arslan Basharat, Matt Turek, Yiliang Xu, Chuck Atkins, David Stoup, Keith Fieldhouse, Paul Tunison, and Anthony Hoogs. Real-time multi-target tracking at 210 megapixels/second in wide area motion imagery. In Winter Conference on Applications of Computer Vision (WACV). IEEE, 2014.

3. [Paper] Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. Probabilistic CFG with latent annotations. Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics ACL 05 , 05pages: 75-82, 2005. doi:doi:10.3115/1219840.1219850. URL http://portal.acm.org/citation.cfm?doid=1219840.1219850.

4. [Paper] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning Accurate, Compact, and Interpretable Tree Annotation. In Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, number July, pages 433_440, 2006.

5. [Paper] Shay B Cohen, Karl Stratos, Michael Collins, Dean P Foster, and Lyle Ungar. Experiments with Spectral Learning of Latent-Variable PCFGs. Journal of Machine Learning Research, 15 (2014), 2399-2449.

7.0 APPENDICES

7.1 Appendix A: Summer School Participant Demographics

The following information was compiled from application data provided by the summer school participants.

Summer School Participation	2017	2016	2015	2014
Applications Received	45	44	56	21
Applications Accepted	42	35	25	21
Participant Count	34	27	23	21

Highest level of education	2017 Participants	2016 Participants	2015 Participants	2014 Participants
Bachelors, in progress	1 (3%)	3 (11%)	4 (17%)	4 (19%)
Bachelors, completed	3 (9%)	3 (11%)	5 (22%)	5 (24%)
Masters, in progress	1 (3%)	2 (7%)	3 (13%)	
Masters, completed	6 (18%)	8 (30%)	3 (13%)	
Doctorate, in progress	14 (41%)	8 (30%)	6 (26%)	9 (43%)
Doctorate, completed	9 (26%)	3 (11%)	2 (9%)	3 (14%)
Total	**34 (100%)**	**27 (100%)**	**23 (100%)**	**21 (100%)**

Organization Type	2017 Participants	2016 Participants	2015 Participants	2014 Participants
Academic	16 (47%)	12 (44%)	9 (39%)	15 (71%)
Commercial	18 (53%)	15 (56%)	14 (61%)	6 (29%)
Total	**34 (100%)**	**27 (100%)**	**23 (100%)**	**21 (100%)**

Field of Education	2017 Participants	2016 Participants	2015 Participants	2014 Participants
Artificial Intelligence / Machine learning	3 (9%)	3 (11%)		
Astronomy			1 (4%)	
Biology				1 (5%)
Chemistry			1 (4%)	
Computational Science[3]	1 (3%)	1 (4%)		
Criminal Justice	1 (3%)			
Economics	1 (%)	1 (4%)	2 (9%)	
Engineering [1]	1 (3%)	2 (7%)	4 (17%)	2 (10%)
Finance	1 (3%)			
Genetics	1 (3%)			
Informatics/Data Science	2 (6%)		1 (4%)	3 (14%)
Math & Computer Science	18 (53%)	16 (59%)	11 (48%)	12 (57%)
MBA	2 (6%)			
Neuroscience			1 (4%)	1 (5%)
Other/Not Specified	1 (3%)		1 (4%)	
Philosophy				1 (5%)
Physics [2]	1 (3%)	2 (7%)	1 (4%)	1 (5%)
Political Science	1 (3%)			
Psychology		1 (4%)		
Robotics		1 (4%)		
Total	**34 (100%)**	**27 (100%)**	**23 (100%)**	**21 (100%)**

[1] Includes Aerospace, Biomedical, Electrical, Biological, Mechanical, Chemical
[2] Includes Biophysics
[3] Includes Social, Hydrodynamics

LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

AFRL	Air Force Research Laboratory
ARPA	Activity Recognition and Proactive Assistance
BP	Belief Propogation
CALO	Cognitive Assistant that Learns and Organizes
CDC	Centers for Disease Control and Prevention
CFG	Context-free Grammar
CP	Challenge Problem
CRA	Charles River Analytics
DARPA	Defense Advanced Research Projects Agency
DoD	Department of Defense
HMC	Hamiltonian Monte-Carlo
ILI	Influenza-like Illness
ISR	Intelligence, Surveillance, and Reconnaissance
LDC	Linguistic Data Consortium
MAP	Maximum Aposteriori Probability
MCMC	Markov chain Monte Carlo
MIDAS	Multimedia Digital Archiving System
MIT	Massachusetts Institute of Technology
N	Noun
NLP	Natural Language Processing
NP	Noun Phrase
PCFG	Probabilistic Context-free Grammar
PI	Principal Investigator
PPAML	Probabilistic Programming for Advancing Machine Learning
PPL	Probabilistic Programming Language
PPS	Probabilistic Programming System
RFP	Request for Proposal
TA	Technical Area
TCP	Team Challenge Problem
UAS	Unmanned Autonomous Systems
WAMI	Wide Area Motion Imagery
WPAFB	Wright Patterson Air Force Base
WSJ	Wall Street Journal